HSF Detector Simulations WG Meeting Live Notes 7th June 2021

Please add comments questions including your name, e.g.
- *This is an example question [Ben Morgan]*

Introduction (Krzysztof Genser)

PyG4ometry (Stewart Boogert)

- How detailed is the Python binding(s) to Geant4? If G4 supplied a “better” Python binding, what would you need? [Ben Morgan]
    - No actual bindings as such! Lightweight classes that just provide the API [Stewart Boogert]
- How easy to implement import of another GDML-like language, e.g. CMS [Ianna Osboune]?
    - Should be straightforward as XML, but there can be problems of scale [Stewart Boogert]
- Slide 33 on tessellated data. In XML could use the “entity” concept to separate out the tessellated part? [Witek Pokorski]
    - Yes, though could have large files. For certain things may want to load a separate format file [Stewart Boogert]
- On VTK, how does it compare to tools that HEP are using [Ben Couterier]
    - It’s really great for scientific visualization, think VTK as “Geant4”, Paraview as “G4 App”. Particularly useful/industry standard for medical physics. [Stewart Boogert].
- Is the support for CAD STEP/STL based on what the packages/users have used, or a limitation? E.g. Material formats? [Gerardo Ganis]
    - Basically what we use/familiar with. Material formats are an ongoing issue. Probably just use GDML, as mostly just want vis/etc [Stewart Boogert]
- Offline request to expand discussion of CAD conversion and usage [Krzysztof Genser]
    - So loading CAD models is usually performed by a tessellation step and this can result is
        - needles or caps. Sharp triangles or wide angle triangles.
        - even worse non-closed, non-manifold tessellated meshes
        - geometry which is burdensome for Monte Carlo packages
        - Materials are not well described in STEP and there isn’t something standard like auxiliary information that can be added to parts or solids in STEP or other cad formats.

- So pg4ometry can also interface to packages which tidy up and rationalise the meshes, e.g an interface to CGAL, gmsh or tetgen. This means the user can have much more control over this approximation step
- There are algorithms which decompose CAD bodies into Geant4 solids and perform a union of those solids. There are a couple of good examples I've seen in the wild from the neutron/fusion community. There has not been much adoption in the HEP community. These are fundamentally decomposition algorithms based on the non-convex solids or when curved surfaces are involved. I am seriously considering adding this functionality to py4geometry as I have all the ingredients (ability to load and manipulate the appropriate files) but time. It might require a specific CAD solid in Geant4 but then what about acceleration plans using VecGeom.
    - Improved algorithms and advanced features of the CAD to MC conversion tool McCad, FusionEngineeringandDesign 89 (2014)1885–1888
    - Improved solid decomposition algorithms for the CAD-to-MC conversion tool McCad, Fusion Engineering and Design 124 (2017) 1269–1272
    - CMGC: a CAD to Monte Carlo geometry conversion code, NUCL SCI TECH (2020) 31:82
- Materials. One possible way is to key materials using the body name in STEP and then using a package like pyg4ometry//DD4Hep interrogate the name to implement the material in the gdml. This is clearly not satisfactory, but might be the best path given limited time and resources. So for example a solid in CAD is given the name
    - Solidname_material_visatt
- Material and visatt are some keys to a dictionary of materials and visualisation attributes which are implemented by particle physicists opposed to the design engineers. Something similar can be done with STEP comment fields. [Stewart Boogert, Laurie Nevay]
- There are lots of other issues regarding CAD, here is a list : parts/assemblies, compound solids… optical surfaces, other auxiliary properties needed for simulation.

GDML: Status and Update (Witold Pokorski)

- With projects like ROOT, Geant4 etc having their own parsers, is there scope for a single parser library [Ben Morgan]
    - Actually done/considered in the early days, but ended up discarding. Found that XML parsing was very simple to implement, so straightforward, and easier, for projects to implement their own direct reader [Witek Pokorski]
- Are there any recommendations for checking "round-tripping", e.g. write out GDML from ROOT, read elsewhere, to check the geometry is the same? [Ianna Osbourne]
    - No real suggestion on "good practices" here, other than to check visually or for overlaps [Witek] Could also ray trace using Geantinos, at least in Geant4 [Gabriele Cosmo]

    - Other checks would be per tool, or context sensitive, e.g. walking geometry tree [Andrei Gheata]
- Several packages that attempt to be an “authoritative” source, so what would the direction be here, if any? [Krzysztof Genser]
    - GDML itself is just the language/schema - don’t see an overlap here with DD4Hep or pyg4ometry [Witek Pokorski]
- On tools to check geometry, could be quite useful to have these separately from the implementations like G4/ROOT. Use case here could be CI etc [Ben Couturier]
    - Wouldn’t G4 be the “tool” in this case? [Witek Pokorski]
    - We had issues with some tags like assemblies, found some losses between pure GDML -> Geant4 [Ben Couturier]
    - That sounds like a parser issue, but can discuss offline [Witek Pokorski]
    - The issue is how to define what a “tool(s)” would do and if that fits all use cases - may be simpler for projects to implement their own. Open to discussion though! [Witek Pokorski]